U.S. DEPARTMENT OF COMMERCE • National Institute of Standards and Technology

NISTIR 88-4022

National Computer Systems Laboratory

ISSUES IN INTERPRETING THE EXPORT ADMINISTRATION REGULATIONS' "PROCESSING DATA RATE"

Vivian Lawrence

U. S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Advanced Systems Division
Gaithersburg, MD 20899

CMRF

COMPUTER MEASUREMENT RESEARCH FACILITY FOR HIGH PERFORMANCE PARALLEL COMPUTATION

February 1989

Sponsored by the
Bureau of Export Administration
Department of Commerce
14th and Constitution Avenue
Washington, D.C. 20230

ISSUES IN INTERPRETING THE EXPORT ADMINISTRATION REGULATIONS' "PROCESSING DATA RATE"

Vivian Lawrence

Advanced Systems Division
National Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

Sponsored by the
Bureau of Export Administration
Department of Commerce

U.S. Department of Commerce, Robert A. Mosbacher, Secretary

Ernest Ambler, Acting Under Secretary for Technology

National Institute of Standards and Technology
Raymond G. Kammer, Acting Director

February 1989

TABLE OF CONTENTS

ISSUES IN INTERPRETING THE EXPORT ADMINISTRATION REGULATIONS' "PROCESSING DATA RATE"

Vivian Lawrence

The Processing Data Rate (PDR) is one of several criteria used to control export of computer systems. New and varied computer architectures make consistent interpretation of the definitions to be applied in computing the PDR difficult. This report identifies those aspects of the definitions for which interpretation may be difficult, and reviews a draft "PDR Handbook" which attempts to provide an interpretation methodology.

Key words: bit processing rate; computers; export; PDR; performance; processing data rate.

INTRODUCTION

To adequately protect concerns of national importance, and at the same time equitably allow the export of non-critical technologies, a figure of merit for a computer system's computational capability -- referred to herein as the Processing Data Rate (PDR) -- has been one of several criteria upon which export control decisions are based. The PDR is essentially a bit processing rate, taking into account the fixed and floating point arithmetic capabilities of a given system. The definitions necessary for the computation of the PDR have evolved over the past 15-20 years, and are contained in Advisory Note 16 of Export Control Commodity List Number (ECCN) 1565A of the Export Administration Regulations [EAR87].

Unfortunately, the variety of computer architectures currently available and under development makes a single, concise interpretation of the PDR definitions difficult. Neither would it be a simple task to redefine a good single performance measurement. Nevertheless, recent disagreement over the appropriate interpretation of these definitions, and consequent variation in PDR values that might be assigned to a given computer, has led to an attempt to clarify the requirements for the PDR calculation through an explanatory interpretation with examples. Toward this end, the Department of Defense has recently made available a "PDR Handbook" [DoD88].

This report identifies the major issues to be addressed in such an interpretation and, with these in mind, reviews the draft PDR Handbook.

Throughout this report, italicized words and phrases refer to concepts defined in [EAR87]. Displayed definitions are likewise from [EAR87], unless otherwise attributed. Comments on the contents of the PDR Handbook refer to the version dated June, 1988 [DoD88]. The Figures from this version were unavailable, however, so references to Figures herein are to those in the PDR Handbook dated September, 1986 [DoD86].

OVERVIEW OF THE DEFINITIONS

This section provides brief general descriptions of the relevant definitions from [EAR87].

The *processing data rate* of a central processing unit is defined to be the maximum of its *fixed point processing data rate* and its *floating point processing data rate.* If there are no fixed or floating point addition or multiplication instructions, then the *processing data rate* is zero. The definitions of *total processing data rate* and *cumulative total processing data rate* provide formulas for combining the *processing data rates* of multiple central processing units.

The *fixed* and *floating point processing data rates* are computed by dividing an unnormalized weighted average of the number of bits in an instruction and in an operand by the execution time.

The definition of *number of bits in an instruction* restricts consideration to those instructions that permit "full direct addressing of *main storage*," and provides directions for cases when multiple instructions are required to simulate an appropriate single instruction, and when a base register is used to expand the addressing capability of an instruction.

The definitions of *number of bits in a fixed point operand* and *number of bits in a floating point operand* specify, for fixed and floating point operations, that the shortest available operand length (subject to a certain minimum in each case) be used.

The definition of *execution time* specifies that the instruction to be timed is the fastest satisfying restrictions on the addressing mode, the locations of the instruction and operands, and the destination of the result; it provides formulas for computing a single *execution time* when a range of times is given; and it provides instructions for the cases when multiple instructions are required to simulate an appropriate single instruction, and when a base register is used to expand the addressing capability of an instruction.

The definitions of *main storage* and *most immediate storage* provide the basis for determining the location of the instruction and one operand (as directed by the definition of *execution time*) and the requisite addressing capability of an instruction (according to the definition of *number of bits in an instruction*).

COMMENTS ON THE DEFINITIONS AND THEIR INTERPRETATION

Definition of Most Immediate Storage

Most immediate storage is of importance because it is taken to be the location of the instruction and the second operand in determining the execution times of operations. The definition of *most* *immediate* storage depends on that of *main storage*, which is:

> The primary storage for data or instructions for rapid access by a central processing unit. It consists of the internal storage of a *digital computer* and any hierarchical extension thereto, such as cache storage or non-sequentially accessed extended storage.

Most immediate storage is then defined as:

> The portion of the *main storage* most directly accessible by the central processing unit:
>
> (a) For single level *main storage*, this is the internal storage; or
>
> (b) For hierarchical *main storage*, this is:
>
> (1) The cache storage;
>
> (2) The instruction stack; or
>
> (3) The data stack.

It is important to define the extent of *main storage*, particularly addressing the possible inclusion of storage locations that are most directly accessible by the central processing unit. In particular, an interpretation is needed as to whether program-addressable registers and/or internal registers (or possibly only some internal registers) are to be considered *most immediate storage*.

[DoD88] suggests that the definition of *main storage* be interpreted "in a manner consistent with the accepted [International Organization for Standardization] ISO definition of 'main storage'":

> Program addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing.

Several potential problems arise if this suggestion is to be taken into account in interpreting the definitions in [EAR87]:

Instruction and data stacks (called out in the definition of *most immediate storage* in [EAR87]) are not always program addressable.

It is not clear whether the ISO definition means to exclude all registers from consideration as "main storage," whether it means to exclude registers functioning as accumulators, or whether program addressability is meant to be the primary criterion for inclusion. If registers are never to be considered *main storage*, then machines with extensive register structures in lieu of cache might have low *processing data rates* that do not necessarily reflect their apparent processing capabilities. On the other hand, if some registers are to be considered part of *main storage*, then clarification of the above issues is necessary because, for example, registers functioning as accumulators can be program addressable.

Section 3 of [DoD88], which provides a technical discussion of the definitions in [EAR87], seems to favor the inclusion of program addressable registers, including those "from which data can be operated on." In contrast to section 3, Example 5 of section 4 of [DoD88] describes a machine with register-to-register instruction formats in which the two register operand locations are specified. Despite the fact that at least some of the registers are therefore program addressable, registers are not taken to be *most immediate storage.*

In Example 6 of [DoD88], however, the instruction registers are considered to be part of *main storage* (and therefore *most immediate storage*). It is explained that executed instructions are not necessarily pushed off the stack maintained in those registers, but sometimes remain and are program addressable, for example when they occur in a loop. If it is the case that non-loop instructions do not remain on the instruction stack, then this is an interpretation that could be made the other way with equal ease. Such an interpretation should be accompanied by a more complete description of the system's capabilities, and justified in terms of general principles that would provide guidance for related interpretations.

Finally, considering program addressability in determining the extent of *main storage* could lead to an incompatible set of numbers being used in the calculation of the *processing data rate*: If registers are considered to be *most immediate storage*, then *execution time* will be computed using register-to-register operations. However, the *number of bits in an instruction, fixed or floating point, addition or multiplication* is defined to be:

> The appropriate shortest single fixed or floating point instruction length that permits full direct addressing of the *main storage.*

Thus, the number of bits in a longer instruction would be counted, but the execution time would be that of a shorter, faster instruction. The issue of whether the same instruction should be used in computing *execution time* and the *number of bits in an instruction* is discussed further in the section entitled "Choice of Instructions" below. It should be noted that [DoD88] consistently uses the execution time for the instruction chosen.

Consideration of Concurrent Processing Capabilities

Concurrent processing capabilities can be a result of a variety of design features. These include Multiple Instruction Stream/Multiple Data Stream, or MIMD, architectures (with or without a shared memory); processors with attached coprocessors (whose purpose is to execute some subset of the instructions provided to the main processor); Single Instruction Stream/Multiple Data Stream, or SIMD, architectures; and pipelining.

The possible concurrency provided by such architectures has substantial bearing on the performance capabilities of the machines, and should be adequately reflected in the *total processing data rate.* An interpretation of the appropriate definitions is needed to ensure that the rates associated with computers having these features provide accurate relative measures of their capabilities. In their current form, the definitions do not adequately address the architectures currently available.

The notion that export evaluation should assume the maximum possible concurrency was discussed in [LYO88] in the context of multiple processors, and is consonant with the treatment of instruction prefetch in the examples in [DoD88] (see discussion of pipelining below). It also follows what appears to be the philosophy behind the definition of *execution time* in [EAR87], in which the instruction and one operand are taken to be in *most immediate storage*, with no derating applied for the cache hit rate in the case when cache is *most immediate storage*. This approach should be considered in developing interpretations to reflect the effects of concurrency.

MIMD Architectures. [EAR87] provides instructions for taking into account the presence of multiple processors or of multiple functional units within a single processor in the definition of *total processing data rate*:

> (a) Of a single central processing unit, is its *processing data rate*;
>
> (b) Of multiple central processing units that do not share direct access to a common *main storage*, is:
>
> > The individual *processing data rate* of each central processing unit, i.e., each unit is separately treated as a single central processing unit as in (a) above; or
>
> (c) Of multiple central processing units that partially or fully share direct access to a common *main storage* at any level is the sum of:
>
> > (1) The highest of the individual *processing data rates* of all central processing units; and
> >
> > (2) 0.75 times the *processing data rate* of each remaining central processing unit, sharing the same *main storage*;
> >
> > assuming the configuration of equipment that would maximize this sum of rates.

The definition of *total procesing data rate* could be usefully augmented by describing the motivation behind it. Such motivation would provide a needed basis for interpreting the definition when the architecture in question does not clearly fit into case (b) or case (c) above. For example, consider a machine in which each processor has it own "local" memory -- that is, memory it can access without use of the interconnection network linking it to the other processors -- but in which any processor can directly access any other processor's memory, albeit in a somewhat longer period of time. A written record of the motivation behind the definition of *total processing data rate* would provide the most general basis for interpreting that definition for this example and for other architectures.

[DoD88] introduces, without resolving, a new issue which could require interpretation: The statement in section 3.1 that the formula for the *total processing data rate* for multiple central processing units sharing direct access to a common *main storage* assumes "that the memory speed and bus rate are adequate to support full-speed operation of all attached CPUs" provides arguable grounds for derating based on inadequate memory and bus speed. No example is given, however, of when and how this might appropriately be done. Moreover, the definitions in [EAR87] do not appear to provide any basis for such derating.

The definition of *total processing data rate* in case (b) above specifies that multiple central processing units are to be treated individually, with no regard for any additional capabilities the system derives by virtue of their interconnections. The problem is somewhat mitigated by the definition of *cumulative total processing data rate*:

The sum of all *total processing data rates* in a given transaction.

Unfortunately, the *cumulative total processing data rate* is not a criterion in all export control determinations of ECCN 1565A, allowing in some cases relatively large numbers of processors to be exported as one machine, while for other configurations involving fewer processors export would be denied.

Example: (This example is based on the criteria set forth in Advisory Note 17 of [EAR87], which has since been revised. It is included as an indication of the sorts of problems inherent in the definitions as they now read.) Advisory Note 17 of ECCN 1565A [EAR87], which controls exports to the People's Republic of China, specifies that the following limits not be exceeded if an export license is to be approved:

Central processing unit - *main storage* combinations, either

(i) *Total processing data rate* - 155 Mbit per second and *total connected capacity* of *main storage* - 72 Mbit; or

(ii) *Total processing data rate* - 100 Mbits per second and *total connected capacity* of *main storage* - 134.5 Mbit.

Consider the following two configurations:

Configuration 1 is a sixteen node hypercube in which each node consists of a processor with *processing data rate* equal to 88 million bits per second and eight megabits of local memory. With each processor treated separately, the limits of 100 million bits per second for *total processing data rate* and 134.5 million bits for *total connected capacity* of *main storage* are not exceeded. Thus, an export license would not be precluded.

Configuration 2 comprises only two processors identical to those used above, but connected instead to an 80 megabit shared memory. In this case, the *total processing data rate* is greater than 100 million bits per second, and the *total connected capacity* of *main storage* exceeds 72 million bits, so an export license would not be approved.

Thus, due to the failure of Advisory Note 17 to consider the *cumulative total processing data rate*, the configuration with substantially greater ability to do useful computation would be exportable, while the less capable system would not.

Attached Coprocessors. In Example 1 of [DoD88], a microprocessor and its floating point coprocessor are treated as a single CPU because they share a common instruction stream. According to the information given in the example, it is possible that a mix of fixed and floating point operations could be accomplished with some concurrency. If the maximum possible amount of concurrency is to be considered, guidance should be given for accounting for the differences in execution times between a processor/coprocessor pair in which no parallel execution is possible and one in which parallel asynchronous operation can occur.

SIMD Architectures. It is not clear how to interpret the definitions in [EAR87] in the case of a SIMD machine, or whether it would be more appropriate to augment, rather that interpret, these definitions. Given a typical configuration, with a control unit which will fetch, decode, and broadcast instructions, and multiple processing elements which find the needed operands in their own memories, as opposed to in that of the control unit, the definition of *execution time* requiring both the instruction and second operand to be in *most immediate storage* makes no sense. It is also unclear whether the individual processing elements should be considered to be "central processing units," in light of the functionality taken over by the control unit. Finally, the lock-step, replicated nature of computation in these machines, interspersed with communication among processing elements, might call for yet another definition of *total processing data rate*, beyond those for the cases of central processing units with and without access to a common *main storage.*

SIMD architectures, in which a single instruction is executed on multiple data, need not be implemented with massive, or even moderate, parallelism. For example, a single central processing unit might possess the capability to simultaneously execute a single arithmetic operation on several distinct pairs of operands. Such capability is not explicitly treated in [EAR87], nor is it discussed in [DoD88]. A distinction might appropriately be drawn between this case and the large scale parallelism of SIMD machines in which the communication among processing elements mentioned above is a major performance factor.

Pipelining. In cases where an architecture permits an instruction to be prefetched during the execution of a previous instruction, as in Examples 1, 2, and 4, [DoD88] computes execution time assuming the instruction has been prefetched. Thus, the startup cost (the time for the first instruction fetch) is effectively amortized over infinitely many operations, and so adds nothing to the *execution time.* This interpretation is consistent with the notion of assuming maximum possible concurrency.

Example 6 depicts the pipelining of arithmetic operations, and handles the question of startup costs somewhat differently. For the fixed point add instruction, for example, the pipeline is implemented by using different register sets for each iteration of the operation. The number of iterations to be pipelined is effectively taken to be limited by the number of register sets; the startup cost is amortized over this number of operations. The sequence is shown in the timing analysis of Figure 4-25 of [DoD86] as restarting one clock cycle after the read of the operand from the last register set, but it would appear that the sequence could restart one clock cycle after the read of the operand in the first set of registers, recycling through the registers as they become available. Performing the analysis this way assumes maximum possible concurrency, and decreases the *execution time* computed in [DoD88] by 25%.

Choice of Instructions

The choice of instructions on which to base the computation of *processing data rate* is constrained by the definitions of *number of bits in an instruction* and *execution time.*

The *number of bits in an instruction, fixed or floating point, multiplication or addition*, is defined to be:

The appropriate shortest single fixed or floating point instruction length that permits full direct addressing of the *main storage.*

The following note is added:

> If the addressing capability of an instruction is expanded by using a base register, then the *number of bits in an instruction, fixed or floating point, addition or multiplication* is the number of bits in the instruction with the standard address length including the number of bits necessary to use the base register.

The definition of *execution time* contains the following:

> (a) The time certified or openly published by the manufacturer for the execution of the fastest appropriate instruction, under the following conditions:
>
> > (1) No indexing or indirect operations are included;
> >
> > (2) The instruction is in the *most immediate storage*;
> >
> > (3) One operand is in the accumulator or in a location of the *most immediate storage* that is acting as the accumulator;
> >
> > (4) The second operand is in the *most immediate storage*; and
> >
> > (5) The result is left in the accumulator or the same location in the *most immediate storage* that is acting as the accumulator;

The following note is added:

> If the addressing capability of an instruction is expanded by using a base register, then the *execution time* shall include the time for adding the content of the base register to the address part of the instruction.

It should be noted that [EAR87] does not explicitly specify that the same instruction format should be used in computing the *number of bits in an instruction* and the *execution time*. Indeed, when there are several appropriate instructions, the definition of *number of bits in an instruction* requires using the operation of shortest length, while the definition of *execution time* requires the fastest; these may not always be the same. [DoD88] consistently uses the same instruction for both computations. The extent to which this must be done should be made explicit.

Use of Base Registers

The use of base registers to extend the addressing capability of an instruction is specifically mentioned in both the definitions cited in the previous section. Two problems arise in this context:

The distinction between addressing using a base register and addressing using indexing, which is proscribed by the definition of *execution time*, should be spelled out. In some machines, index registers can be used in the same way as base registers.

The notion of the number of bits necessary to use a base register requires explication. Possible interpretations include combinations of the number of bits required for its specification and/or the number of bits required to change the value it contains, and the number of bits it actually contains. The full address length generated as a result of using the base register is yet another possibility. A notion of the motivation behind this definition would help in developing an appropriate interpretation. If, on the one hand, it is meant to describe the number of bits processed in accessing an address, then it might be interpreted to include the number of bits in the base register specification, as well as the number of bits in the base register. On the other hand, if it is meant to reflect the size of *main storage*, then it might appropriately be interpreted as the length of the computed address.

In Example 1 of [DoD88], the instructions to account for the use of a base register are ignored. In this example, full direct addressing of all of *main storage* would require 20 bits, but the computation of *number of bits in an instruction* accounts for only 16 address bits. These 16 bits allow direct addressing only within a contiguous segment of memory; the use of a base register would be necessary to realize the full 20 bit addressing capability. The *execution time*, similarly, does not include the time for adding the contents of the base register to the address part of the instruction. [DoD88] justifies these choices by making the assumption that operands are found in the same memory segment. This claim is unjustified by the information given in the example.

This issue arises also in Example 5 of [DoD88]. In this example, the only appropriate instruction involves the use of a base register, and the number of bits to account for that use is taken to be the number of bits to specify that register.

The issue is further complicated by the following note, appended to the definition of *number of bits in an instruction*:

> When multiple instructions are required to simulate an appropriate single instruction, the [*number of bits in an instruction*] is defined as 16 bits plus the number of bits ... that permits full direct addressing of the *main storage*.

In example 4, the *number of bits in a fixed point instruction* includes 12 bits of mode and displacement used for addressing in the fixed point memory-to-register arithmetic instructions. All floating point operations are, however, implemented as stack operations, requiring the operands first to be loaded onto the stacks. Since two instructions are required, the above note is invoked, and the *number of bits in a floating point instruction* is taken to be 16 bits plus the number of bits that permits full direct addressing of the *main storage*. Despite the fact that 12 bits of mode and displacement were adequate for addressing in the fixed point case, [DoD88] claims that addressing, through the use of base registers, requires 16 bits in this case.

(One of the examples in [DoD88] contains an obvious error: Example 3 describes a system that can be configured with a standard memory addressable with 15 bits, or with expanded memory requiring a 20 bit address, generated through the use of a base register. The instruction formats considered to calculate the *number of bits in an instruction* use the 15 bit address, but the *execution times* are taken assuming the extended memory.)

Number of Bits in an Operand

The *number of bits in a fixed point operand* and the *number of bits in a floating point operand* are used to determine the bit processing rates of fixed and floating point arithmetic operations.

The following definitions are made in [EAR87]:

Number of bits in a fixed point operand:

> (a) The shortest fixed point operand length; or
>
> (b) 16 bit;
>
> whichever is greater.

Number of bits in a floating point operand:

> (a) The shortest floating point operand length; or
>
> (b) 30 bit;
>
> whichever is greater.

The following interpretation is made in Example 5 of [DoD88]: Although 16 bit fixed point operations are supported in this example, such operands are converted internally to 32 bits (because of the internal register structure). [DoD88] requires the *processing data rate* computation to account for all 32 bits. This might equally reasonably be interpreted the other way, requiring only 16 bits. In either case, the interpretation made should be based on a stated general philosophy, so that other, similar interpretations can be made consistently.

The definitions above place a premium on 16 bit fixed and 30 bit floating point operations and, indeed, ignore any faster computational capabilities a machine may exhibit on longer operands. It should be noted that shorter operand lengths are not the most important for many scientific computations. Thus, a *processing data rate* computed using these definitions is not as reflective of computational capabilities of interest as would be one, say, that took the highest processing data rate over all operand lengths.

Note that the definition of *execution time* contains the following:

> (d) If the longest fixed point operand length is smaller than 16 bit, then use the time required for the fastest available subroutine to simulate a 16 bit fixed point operation,

but no similar instruction is given for floating point operations.

Non-Arithmetic Data Processing

As it is defined in [EAR87], the *total processing data rate* takes no account of non-arithmetic data processing capabilities.

CONCLUSIONS

The Processing Data Rate should ideally provide a relative ranking of the computational capabilities of various computers. Its computation for a given machine should be consistent and predictable, based on the machine's architectural features. As discussed in this report, these goals are difficult to attain. By virtue of the variety of architectures currently available and under development, it is difficult to develop definitions for a single figure of merit, and equally difficult to provide adequate guidelines for the interpretation of such definitions.

The definitions relevant to the computation of the Processing Data Rate, as they appear in [EAR87], are inadequate in light of the current technology; that is, there are commonly-used architectures for which they appear not to provide a meaningful result. Further, they have shortcomings in expressing computational capabilities that are of interest, and probably should be of concern in the export control arena.

Given the definitions as they are now written, however, interpretations are needed to match definitions to modern architectures, and to remove vagueness and possible contradictions from the definitions. [DoD88] contributes six examples, which illustrate some of the simpler instructions from [EAR87] and which address some, but not all, of the difficult issues identified in this report. There is inadequate general discussion of how the definitions in question are to be interpreted. In some cases the interpretations given appear to be inconsistent, and questionable interpretations are made with inadequate justification. Nevertheless, these examples are a good start and could be expanded upon to provide much needed guidance.

Recommendations

1. The definitions of [EAR87] need to be revised to meaningfully reflect the computational capabilities of machines with the architectural features identified in this report. If such revisions are not made, interpretations should be written to clarify the scope of applicability of these definitions.

2. Changes to the definitions or interpretations thereof are necessary to remove or clarify apparent inconsistencies.

3. Consideration should be given to changing the definitions to reflect a wider range of computational capabilities.

4. Minor inconsistencies in the presentation of [DoD88] should be corrected.

5. [DoD88] should provide a general discussion of the suggested appropriate interpretations of the definitions, rather than relying primarily on an incomplete set of examples. As discussed in this report, the difficulty of providing sufficiently general interpretations might be alleviated by describing the motivations behind the definitions and interpretations; the recording of such motivations would at least provide a basis for future revisions or extensions that are consistent with current use.

Acknowledgments

This report benefited from the author's extensive discussions with Robert Carpenter and Gordon Lyon.

REFERENCES

[DoD86] "Interagency Computer Export Licensing Data System, Technical Reference Volume I: Processing Data Rate (PDR) Handbook," Interagency Review Draft, September, 1986

[DoD88] "Interagency Computer Export Licensing Data System, Technical Reference Volume I: Processing Data Rate (PDR) Handbook," draft, June, 1988

[EAR87] Export Administration Regulations, ECCN 1565A, October 1, 1987

[LYO88] Lyon, G., ed., "Benchmarks to Supplant Export 'FPDR' Calculations," NBSIR 88-3795, National Bureau of Standards, 1988

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO.	2. Performing Organ. Report No.	3. Publication Date
	NISTIR 88-4022		FEBRUARY 1989

4. TITLE AND SUBTITLE

Issues in Interpreting the Export Administration Regulations' "Processing Data Rate"

5. AUTHOR(S)

Vivian Lawrence

6. PERFORMING ORGANIZATION (*If joint or other than NBS, see instructions*)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (*Street, City, State, ZIP*)

Office of Export Administration
Department of Commerce
Herbert C. Hoover Building
14th and Constitution Avenue, N.W.
Washington, DC 20230

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (*A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here*)

The Processing Data Rate (PDR) is one of several criteria used to control export of computer systems. New and varied computer architectures make consistent interpretation of the definitions to be applied in computing the PDR difficult. This report identifies those aspects of the definitions for which interpretation may be difficult, and reviews a draft "PDR Handbook" which attempts to provide an interpretation methodology.

12. KEY WORDS (*Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons*)

bit processing rate; computers; export; PDR; performance; processing data rate

13. AVAILABILITY

☐ Unlimited
[✓ For Official Distribution. Do Not Release to NTIS]
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

15. Price